Gerrymandering on Graphs: Computational Complexity and Parameterized Algorithms

Sushmita Gupta[1], Pallavi Jain[2], Fahad Panolan[3], Sanjukta Roy[4(✉)], and Saket Saurabh[1]

[1] The Institute of Mathematical Sciences, HBNI, Chennai, India
{sushmitagupta,saket}@imsc.res.in
[2] Indian Institute of Technology Jodhpur, Jodhpur, India
pallavi@iitj.ac.in
[3] Indian Institute of Technology Hyderabad, Hyderabad, India
fahad@cse.iith.ac.in
[4] TU Wien, Vienna, Austria
sanjukta.roy@tuwien.ac.at

Abstract. This paper studies *gerrymandering on graphs* from a computational viewpoint (introduced by Cohen-Zemach et al. [AAMAS 2018] and continued by Ito et al. [AAMAS 2019]). Our contributions are two-fold: conceptual and computational. We propose a generalization of the model studied by Ito et al., where the input consists of a graph on n vertices representing the set of voters, a set of m candidates $\mathcal{C}$, a weight function $w_v : \mathcal{C} \to \mathbb{Z}^+$ for each voter $v \in V(G)$ representing the preference of the voter over the candidates, a distinguished candidate $p \in \mathcal{C}$, and a positive integer k. The objective is to decide if it is possible to partition the vertex set into k *districts* (i.e., pairwise disjoint connected sets) such that the candidate p *wins* more districts than any other candidate. There are several natural parameters associated with the problem: the number of districts (k), the number of voters (n), and the number of candidates (m). The problem is known to be NP-complete even if $k = 2$, $m = 2$, and G is either a complete bipartite graph (in fact $K_{2,n}$, i.e., partitions of size 2 and n) or a complete graph. Moreover, recently we and Bentert et al. [WG 2021], independently, showed that the problem is NP-hard for paths. This means that the search for FPT algorithms needs to focus either on the parameter n, or subclasses of forest (as the problem is NP-complete on $K_{2,n}$, a family of graphs that can be transformed into a forest by deleting *one* vertex). Circumventing these intractability results we successfully obtain the following algorithmic results.

Sushmita Gupta supported by SERB-Starting Research Grant (SRG/2019/001870). Fahad Panolan supported by Seed grant, IIT Hyderabad (SG/IITH/F224/2020-21/SG-79). Sanjukta Roy supported by the WWTF research grant (VRG18-012). Saket Saurabh supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

I. Caragiannis and K. A. Hansen (Eds.): SAGT 2021, LNCS 12885, pp. 140–155, 2021.
https://doi.org/10.1007/978-3-030-85947-3_10

– A $2^n(n+m)^{\mathcal{O}(1)}$ time algorithm on general graphs.
– FPT algorithm with respect to k (an algorithm with running time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$) on paths in both deterministic and randomized settings, even for arbitrary weight functions. Whether the problem is FPT parameterized by k on trees remains an interesting open problem.

Our algorithmic results use sophisticated technical tools such as representative set family and Fast Fourier Transform based polynomial multiplication, and their (possibly first) application to problems arising in social choice theory and/or algorithmic game theory is likely of independent interest to the community.

Keywords: Gerrymandering · Parameterized complexity · Representative set

1 Introduction

"Elections have consequences" a now-famous adage ascribed to Barack Obama, the former President of U.S.A, brings to sharp focus the high stakes of an electoral contest. Political elections, or decision making in a large organization, are often conducted in a hierarchical fashion. Thus, in order to win the final prize it is enough to manipulate at district/division level, obtain enough votes and have the effect propagate upwards to win finally. Needless to say the ramifications of winning and losing are extensive and possibly long-term; consequently, incentives for *manipulation* are rife.

The objective of this article is to study a manipulation or control mechanism, whereby the manipulators are allowed to create the voting "districts". A well-thought strategic division of the voting population may well result in a favored candidate's victory who may not win under normal circumstances. In a more extreme case, this may result in several favored candidates winning multiple seats, as is the case with election to the US House of Representatives, where candidates from various parties compete at the district level to be the elected representative of that district in Congress. This topic has received a lot of attention in recent years under the name of *gerrymandering*. A New York Times article "How computers turned gerrymandering into science" [16] discusses how Republicans were able to successfully win 65% of the available seats in the state assembly of Wisconsin even though the state has about an equal number of Republican and Democrat voters. The possibility for gerrymandering and its consequences have long been known to exist and have been discussed for many decades in the domain of political science, as discussed by Erikson [17] and Issacharoff [23]. Its practical feasibility and long-ranging implications have become a topic of furious public, policy, and legal debate only somewhat recently [33], driven largely by the ubiquity of computer modelling in all aspects of the election process. Thus, it appears that via the vehicle of gerrymandering the political battle lines have been drawn to (re)draw the district lines.

While gerrymandering has been studied in political sciences for long, it is only rather recently that the problem has attracted attention from the perspective of

algorithm design and complexity theory. Lewenberg et al. [26] and Eiben et al. [15] study gerrymandering in a geographical setting in which voters must vote in the closest polling stations and thus problem is about strategic placement of polling stations rather than drawing district lines. Cohen-Zemach et al. [8] modeled gerrymandering using graphs, where vertices represent voters and edges represent some connection (be it familial, professional, or some other kind), and studied the computational complexity of the problem. Ito et al. [24] further extended this study to various classes of graphs, such as paths, trees, complete bipartite graphs, and complete graphs.

In both the papers the following hierarchical voting process is considered: A given set of voters is partitioned into several groups, and each of the groups holds an independent election. From each group, one candidate is elected as a nominee (using the plurality rule). Then, among the elected nominees, the winner is determined by a final voting rule (again by plurality). The formal definition of the problem, termed Gerrymandering (GM), considered in [24] is as follows. The input consists of an undirected graph G, a set of candidates $\mathcal{C}$, an approval function $a : V(G) \to \mathcal{C}$ where $a(v)$ represents the candidate approved by v, a weight function $w\colon V(G) \to \mathbb{Z}^+$, a distinguished candidate p, and a positive integer k. We say a candidate q wins a subset $V' \subseteq V(G)$ if $q \in \arg\max_{q' \in \mathcal{C}} \left\{ \sum_{v \in V',\, a(v)=q'} w(v) \right\}$, i.e., the sum of the weights of voters in the subset V' who approve q is not less than that of any other candidate. The objective is to decide whether there exists a partition of $V(G)$ into k non-empty parts $V_1 \uplus \ldots \uplus V_k$ (called *districts*) such that (i) the induced subgraph $G[V_i]$ is connected for each $i \in \{1, \ldots, k\}$, and (ii) the number of districts *won only by* p is more than the districts won by any other candidate alone or with others.

In this paper we continue the line of investigation done in [8,24]. Our contribution is two fold, conceptual and the other is computational. Towards the former, we offer a realistic generalization of GM, named Weighted Gerrymandering (W-GM). Towards the latter, we present *fixed parameter tractable* (FPT) algorithms with respect to natural parameters associated with the gerrymandering problem.

Our Model. A natural generalization of GM in real-life is that of a vertex representing a locality or an electoral booth as opposed to an individual citizen. In that situation, however, it is natural that more than one candidate receives votes in a voting booth, and the number of such votes may vary arbitrarily. We can model the number of votes each candidate gets in the voting booth corresponding to booth v by a weight function $w_v : \mathcal{C} \to \mathbb{Z}^+$, i.e., the value $w_v(c)$ for any candidate $c \in \mathcal{C}$ represents the number of votes obtained by candidate c in booth v. This model is perhaps best exemplified by a nonpartisan "blanket primary" election (such as in California) where all candidates for the same elected post regardless of political parties, compete on the same ballot against each other all at once. In a two-tier system, multiple winners (possibly more than two) are declared and they contest the general election. The idea that one

can have multiple candidates earning votes from the same locality and possibly emerging as winners is captured by GM. In [8,24], the vertex v "prefers" only one candidate, and in this sense our model (W-GM) generalizes theirs (GM).

Formally stated, the input to W-GM consists of an undirected graph G, a set of candidates $\mathcal{C}$, a weight function for each vertex $v \in V(G)$, $w_v : \mathcal{C} \to \mathbb{Z}^+$, a distinguished candidate p, and a positive integer k. A candidate q is said to win a subset $V' \subseteq V(G)$ if $q \in \arg\max_{q' \in \mathcal{C}} \left\{\sum_{v \in V'} w_v(q')\right\}$. The objective is to decide whether there exists a partition of the vertex set $V(G)$ into k districts such that (i) $G[V_i]$ is connected for each $i \in [k]$, and (ii) the number of districts *won only by* p is more than the number of districts won by any other candidate alone or with others. GM can be formally shown to be a special case of W-GM since we can transform an instance $\mathcal{I} = (G, \mathcal{C}, a, w, p, k)$ of GM to an instance $\mathcal{J} = (G, \mathcal{C}, \{w_v\}_{v \in V(G)}, p, k)$ of W-GM as follows. For each $v \in V(G)$, let $w_v : \mathcal{C} \to \mathbb{Z}^+$ such that for any $q \in \mathcal{C}$, if $a(v) = q$, then $w_v(q) = w(v)$ and $w_v(q) = 0$, otherwise.

Our Results and Methods. The main open problem mentioned in Ito et al. [24] is the complexity status of GM on paths when the number of candidates is not fixed (for the fixed number of candidates, it is solvable in polynomial time). This question was recently resolved by Bentert et al. [1], and has also been proved independently by us, which is presented in our extended version [22] and omitted from here because of lack of space. Thus, in this article we will focus on designing efficient algorithms. We must remark that Bentert et al. [1] also show that the problem is weakly NP-hard for trees with three or more candidates.

We study the problem from the viewpoint of parameterized complexity. The goal of parameterized complexity is to find ways of solving NP-hard problems more efficiently than brute force: here the aim is to restrict the combinatorial explosion in the running time to a parameter that is expected to be much smaller than the input size. Formally, a *parameterization* of a problem is assigning an integer ℓ to each input instance and we say that a parameterized problem is *fixed-parameter tractable* (FPT) if there is an algorithm that solves the problem in time $f(\ell) \cdot |I|^{O(1)}$, where $|I|$ is the size of the input and f is an arbitrary computable function depending on the parameter ℓ only. There is a long list of NP-hard problems that are FPT under various parameterizations. For more background, the reader is referred to the monographs [9,14,29].

Our Choice of Parameters. There are several natural parameters associated with the gerrymandering problem: the number of districts the vertex set needs to be partitioned (k), the number of voters (n), and the number of candidates (m). Ito et al. [24] proved that GM is NP-complete even if $k = 2$, $m = 2$, and G is either a complete bipartite graph (in fact $K_{2,n}$) or a complete graph. Thus, we cannot hope for an algorithm for W-GM that runs in $f(k, m) \cdot n^{\mathcal{O}(1)}$ time, i.e., an FPT algorithm with respect to the parameter $k + m$, even on planar graphs. In fact, we cannot hope to have an algorithm with running time $(n + m)^{f(k,m)}$, where f is a function depending only on k and m, as that would imply P=NP. This means that our search for FPT algorithms needs to either focus on the parameter n, or subclasses of planar graphs (as the problem is NP-complete on $K_{2,n}$, which

is planar). Furthermore, note that $K_{2,n}$ could be transformed into a forest by deleting a vertex, and thus we cannot even hope to have an algorithm with running time $(n+m)^{f(k,m)}$, where f is a function depending only on k and m, on a family of graphs that can be made acyclic, in fact a star, by *deleting at most one vertex*. This essentially implies that if we wish to design an FPT algorithm for W-GM with respect to the parameter k, or m, or $k+m$, we must restrict input graphs to forests. Circumventing these intractable results, we successfully obtain several algorithmic results. We give deterministic and randomised FPT algorithms for W-GM on paths with respect to k. Since W-GM generalizes GM, the algorithmic results hold for GM as well.

Theorem 1. *There is an algorithm that given an instance of* W-GM *on arbitrary graphs and a tie-breaking rule η, solves the instance in time $2^n(n+m)^{\mathcal{O}(1)}$.*

Intuition Behind the Proof of Theorem 1. Suppose that we are given a Yes-instance of the problem. Of the k possibilities, we first "guess" in a solution the number of districts that are won by the distinguished candidate p. Let this number be denoted by $k^\star$. Next, for every candidate $c \in \mathcal{C}$, we consider the family $\mathcal{F}_c$, the set of districts of $V(G)$ in which c wins in each of them. These families are pairwise disjoint because each district has a unique winner. Our goal is to find $k^\star$ disjoint sets from the family $\mathcal{F}_p$ and at most $k^\star - 1$ disjoint sets from any other family so that in total we obtain k pairwise disjoint districts that partition $V(G)$. The exhaustive algorithm to find the districts from these families would take time $\mathcal{O}^\star(2^{nmk^\star})$. We reduce our problem to polynomial multiplication involving polynomial-many multiplicands, each with degree at most $\mathcal{O}(2^n)$.

Why Use Polynomial Algebra? Every district S is a subset of $V(G)$. Let $\chi(S)$ denotes the characteristic vector corresponding to S. We view $\chi(S)$ as an n-digit binary number, in particular, if $u_i \in S$, then i^{th} bit of $\chi(S)$ is 1, otherwise 0. A crucial observation guiding our algorithm is that two sets S_1 and S_2 are disjoint if and only if the number of 1 in $\chi(S_1) + \chi(S_2)$ (binary sum/modulo 2) is equal to $|S_1| + |S_2|$. So, for each set $\mathcal{F}_c$, we make a polynomial $P_c(y)$, where for each set $S \in \mathcal{F}_c$, there is a monomial $y^{\chi(S)}$. Let c_1 and c_2 be two candidates, and for simplicity assume that each set in $\mathcal{F}_{c_1}$ has size exactly s and each set in $\mathcal{F}_{c_2}$ has size exactly t (we do not have such assumption in the formal description of the algorithm). Let $P^\star(y)$ be the polynomial obtained by multiplying $P_{c_1}(y)$ and $P_{c_2}(y)$; and let y^z be a monomial of $P^\star(y)$. Then, the z has exactly $s+t$ ones if and only if "the sets which corresponds to z are disjoint". Thus, the polynomial method allows us to capture disjointness and hence, by multiplying appropriate subparts of polynomial described above, we obtain our result. Furthermore, note that $\chi(S) \in \{0,1\}^n$, throughout the process, correspond to some set in $V(G)$, and hence the decimal representation of the maximum degree of the considered polynomials is upper bounded by 2^n. Hence, the algorithm itself is about applying an $\mathcal{O}(d \log d)$ algorithm to multiply two polynomials of degree d; here $d \leq 2^n$. Thus, we obtain Theorem 1.

Theorem 2. *There is a deterministic algorithm that given an instance of* W-GM *on paths and a tie-breaking rule η solves in time* $2.619^k(n+m)^{\mathcal{O}(1)}$.

Theorem 3. *There is a randomized algorithm that given an instance of* W-GM *on paths and a tie-breaking rule η, solves the instance in time* $2^k(n+m)^{\mathcal{O}(1)}$ *with no false positives and false negatives with probability at most* $1/3$.

Intuition Behind the Proofs of Theorem 2 and 3. Since, the problem is on paths, it boils down to selecting $k-1$ appropriate vertices which divide the path into k subpaths that form the desired districts. This in turn implies that each district can be identified by the leftmost vertex and the rightmost vertex appearing in the district (based on the way vertices appear on the path). Hence, there can be at most $\mathcal{O}(n^2)$ districts in the path graph. Furthermore, since we are on a path, we observe that if we know a district (identified by its leftmost and the rightmost vertices on the path), then we also know the rightmost (and leftmost) vertex of the district adjacent to its left (resp. right). These observations naturally lead us to consider the following graph H: we have a vertex for each possible district and put an edge from a district to another district, if these two districts appear consecutively on the path graph. Thus, we are looking for a path of length k in H such that (a) it covers all the vertices of the input path (this automatically implies that each vertex appears in exactly one district); and (b) the distinguished candidate wins most number of districts. This equivalence allows us to use the rich algorithmic toolkit developed for designing $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm for finding a k-length path in a given graph [5,28,32].

The above tractability result for paths cannot be extended to graphs with pathwidth 2, or graphs with feedback vertex set (a subset of vertices whose deletion transforms the graph into a forest) of size 1, because GM is NP-complete on $K_{2,n}$ when $k=2$ and $|\mathcal{C}|=2$ (see [24]). Note that the pathwidth of graph $K_{2,n}$ is 2 and it has feedback vertex set size 1. For trees, it is easy to obtain a $\mathcal{O}(\binom{n}{k-1})$ time algorithm by "guessing" the $k-1$ edges whose deletion yields the k districts that constitute the solution. However, a $f(k)n^{\mathcal{O}(1)}$ algorithm for trees so far eludes us. Thus, whether the problem is FPT parameterized by k on trees remains an interesting open problem.

Unique Winner vs Multiple Winner: The definition of GM [24] or its generalization W-GM put forward by us does not preclude the possibility of multiple winners in a district. The time complexity stated in Theorems 1 and 2 is achieved when only one winner emerges from each district, a condition that is attainable using a tie-breaking rule. Notably, the algorithms in Theorems 2 and 3 can be modified to handle the case when multiple winners emerge in some district(s) [22].

Additionally, using our parameterized algorithms (Theorems 2 and 3), we can improve over Theorem 1 when the input graph is a path. That is, using Theorems 2 and 3, and the fact that there exists an algorithm for paths that runs in time $\mathcal{O}(\binom{n}{k-1})$, we conclude that for W-GM on paths, there exists a deterministic algorithm that runs in $\max_{1\leq k\leq n}\min\{\binom{n}{k}, 2.619^k\}$ time, and a

randomized algorithm that runs in $\max_{1 \leq k \leq n} \min\{\binom{n}{k}, 2^k\}$ time. Using, standard calculations we can obtain the following result.

Theorem 4. *There is a (randomized) deterministic algorithm that given an instance of* W-GM *on paths and a tie-breaking rule η, solves the instance in time* $(1.708^n(n+m)^{\mathcal{O}(1)})$ $1.894^n(n+m)^{\mathcal{O}(1)}$.

It is worth mentioning that our algorithmic results use sophisticated technical tools from parameterized complexity–representative set family and Fast Fourier transform based polynomial multiplication–that have yielded breakthroughs in improving time complexity of many well-known optimization problems. Thus, their (possibly first) application to problems arising in social choice theory and/or algorithmic game theory is likely of independent interest. Due to the constraints on space, proofs marked by ♣ are deferred to the full version [22].

Related Work. In addition to the result discussed earlier Ito et al. [24] also prove that GM is strongly NP-complete when G is a tree of diameter four; thereby, implying that the problem cannot be solved in pseudo-polynomial time unless P = NP. As GM is a special case of W-GM, each of the hardness results for GM carry onto W-GM. They also exhibit several positive results: GM is solvable in polynomial time on stars (i.e., trees of diameter two) and that the problem can be solved in polynomial time on trees when k is a constant. Moreover, when the number of candidates is a constant, then it is solvable in polynomial time on paths and is solvable in pseudo-polynomial time on trees. The running time of the algorithm on paths is $k^{2^{|\mathcal{C}|}} n^{\mathcal{O}(1)}$, where n is the number of vertices in the input graph and $\mathcal{C}$ is the set of the candidates. Bentert et al. [1] proved GM is NP-hard on paths even if all vertices have unit weights and it is weakly NP-hard on trees even if $|\mathcal{C}| = 2$; and that the problem is polynomial time solvable for trees with diameter three. Prior to these Cohen-Zemach et al. [8] studied GM on graphs. In addition to the papers discussed earlier, there are far too many articles to list on this subject. Some of them are [4,6,7,11,20,25,30,31,34]. Parameterized complexity of manipulation has received extensive attention over the last several years, [2,3,12,18,19] are just a few examples.

2 Preliminaries

For our algorithmic results we define a variant of W-GM that we call Target Weighted Gerrymandering (TW-GM). The input of TW-GM is an instance of W-GM, and a positive integer $k^\star$. The objective is to test whether the vertex set of the input graph can be partitioned into k districts such that the candidate p wins in $k^\star$ districts alone and no other candidate wins in more than $k^\star - 1$ districts. The following simple lemma implies that to design an efficient algorithm for W-GM it is enough to design an efficient algorithm for TW-GM.

Lemma 1. *If there exists an algorithm that given an instance* $(G, \mathcal{C}, \{w_v\}_{v \in V(G)}, p, k, k^\star)$ *of* TW-GM *and a tie-breaking rule η, solves the instance*

in $f(z)$ time, then there exists an algorithm that solves the instance $(G, \mathcal{C}, \{w_v\}_{v \in V(G)}, p, k)$ of W-GM *in $f(z) \cdot k$ time under the tie-breaking rule η.*

Notations and Basic Terminology. In an undirected graph $G = (V, E)$, uv denotes an edge between the vertices u and v which are called the *endpoints* of uv. For a set $X \subseteq V(G)$, $G[X]$ denotes the graph induced on X. We say that set X is connected if $G[X]$ is a connected graph. In a directed graph $G = (V, A)$, we denote an arc (i.e., directed edge) from u to v by $\langle u, v \rangle$, and say that u is an in-neighbor of v and v is an out-neighbor of u. For $x \in V(G)$, $N^-(x) = \{y \in V(G) \colon \langle y, x \rangle \in A(G)\}$. The in-degree (out-degree) of a vertex x in G is the number of in-neighbors (out-neighbors) of x in G. For background on graph theory we refer to [13].

3 FPT Algorithm for General Graphs

We prove Theorem 1 here. Towards that, we use polynomial algebra that carefully keeps track of the number of districts won by each candidate so that nobody wins (if at all possible) more than p. An intuitive idea was presented in the introduction.

Due to Lemma 1 it is sufficient to prove it for TW-GM.

Before we discuss our algorithm, we introduce some notations. The *characteristic vector* of a set $S \subseteq U$, denoted by $\chi(S)$, is an $|U|$-length vector whose i^{th} bit is 1 if $u_i \in S$, otherwise 0. Two binary strings $S_1, S_2 \in \{0,1\}^n$ are said to be disjoint if for each $i \in \{1, \ldots, n\}$, the i^{th} bit of S_1 and S_2 are different. The *Hamming weight* of a binary string S is denoted by $\mathcal{H}(S)$.

Observation 1. *Let S_1 and S_2 be two binary vectors, and let $S = S_1 + S_2$. If $\mathcal{H}(S) = \mathcal{H}(S_1) + \mathcal{H}(S_2)$, then S_1 and S_2 are disjoint binary vectors.*

Proposition 1 [10]**.** *Let $S = S_1 \cup S_2$, where S_1 and S_2 are two disjoint subsets of the set $V = \{v_1, \ldots, v_n\}$. Then, $\chi(S) = \chi(S_1) + \chi(S_2)$ and $\mathcal{H}(\chi(S)) = \mathcal{H}(\chi(S_1)) + \mathcal{H}(\chi(S_2)) = |S_1| + |S_2|$.*

A monomial x^i, where i is a binary vector, is said to have Hamming weight h, if i has Hamming weight h. The *Hamming projection* of a polynomial $P(x)$ to h, denoted by $\mathcal{H}_h(P(x))$, is the sum of all the monomials of $P(x)$ which have Hamming weight h. We define the representative polynomial of $P(x)$, denoted by $\mathcal{R}(P(x))$, as the sum of all the monomials that have non-zero coefficient in $P(x)$ but have coefficient 1 in $\mathcal{R}(P(x))$, i.e., it only remembers whether the coefficient is non-zero. We say that $P(x)$ *contains the monomial* x^i if its coefficient in $P(x)$ is non-zero. In the zero polynomial, the coefficient of each monomial is 0.

Algorithm. Let $I = (G, \mathcal{C}, \{w_v\}_{v \in V(G)}, p, k, k^\star)$ be an instance of TW-GM. We assume that $k^\star \geq 1$, otherwise $k = 1$ and it is a trivial instance.

For each candidate c_i in $\mathcal{C}$, we construct a family $\mathcal{F}_i$ that contains all possible districts won by c_i. Due to the application of tie-breaking rule, we may assume

that every district has a unique winner. Without loss of generality, let $c_1 = p$, the distinguished candidate. Note that we want to find a family $\mathcal{S}$ of k districts, that contains $k^\star$ elements of the family $\mathcal{F}_1$ and at most $k^\star - 1$ elements from each of the other family $\mathcal{F}_i$, where $i > 1$. The union of these districts gives $V(G)$ and any two districts in $\mathcal{S}$ are pairwise disjoint. To find such k districts, we use the method of polynomial multiplication appropriately using the next proposition. Due to Observation 1 and Proposition 1, we know that subsets S_1 and S_2 are disjoint if and only if the Hamming weight of the monomial $y^{\chi(S_1)+\chi(S_2)}$ is $|S_1| + |S_2|$. Here, degree of a polynomial is the decimal representation of its exponent.

Proposition 2 [27]. *There exists an algorithm that multiplies two polynomials of degree d in $\mathcal{O}(d \log d)$ time.*

For every $i \in \{1, \ldots, m\}$, $\ell \in \{1, \ldots, n\}$, if $\mathcal{F}_i$ has a set of size ℓ, then we construct a polynomial $P_i^\ell(y) = \sum_{\substack{Y \in \mathcal{F}_i \\ |Y|=\ell}} y^{\chi(Y)}$. Next, using polynomials $P_1^\ell(y)$, where $\ell \in \{1, \ldots, n\}$, we will create a sequence of polynomials $Q_{1,j}^s$, where $j \in \{1, \ldots, k^\star - 1\}$, $s \in \{j+1, \ldots, n\}$, in the increasing order of j, such that every monomial in the polynomial $Q_{1,j}^s$ has Hamming weight s. For $j = 1$, we construct $Q_{1,1}^s$ by summing all the polynomials obtained by multiplying $P_1^{s'}$ and $P_1^{s''}$, for all possible values of $s', s'' \in \{1, \ldots, n\}$ such that $s' + s'' = s$, and then by taking the representative polynomial of its Hamming projection to s. If $Q_{1,1}^s$ contains a monomial x^t, then there exists a set $S \subseteq V(G)$ of size s such that $t = \chi(S)$ and S is formed by the union of two districts won by c_1. Next, for $j \in \{2, \ldots, k^\star - 1\}$ and $s \in \{j+1, \ldots, n\}$, we create the polynomial $Q_{1,j}^s$ similarly, using $Q_{1,(j-1)}^{s''}$ in place of $P_1^{s''}$. Formally,

$$Q_{1,1}^s = \mathcal{R}\Big(\mathcal{H}_s\Big(\sum_{\substack{1 \le s', s'' \le s \\ s'+s''=s}} P_1^{s'} \times P_1^{s''}\Big)\Big),\ Q_{1,j}^s = \mathcal{R}\Big(\mathcal{H}_s\Big(\sum_{\substack{1 \le s', s'' \le s \\ s'+s''=s}} P_1^{s'} \times Q_{1,(j-1)}^{s''}\Big)\Big).$$

Thus, if $Q_{1,j}^s$ contains a monomial x^t, then there exists a set $S \subseteq V(G)$ of size s such that $t = \chi(S)$ and S is formed by the union of $j+1$ districts won by c_1. In this manner, we can keep track of the number of districts won by c_1. Next, we will take account of the wins of the other candidates.

Towards this we create a family of polynomials $\mathcal{T} = \{T_{k^\star}, \ldots, T_k\}$ such that the polynomial $T_{k^\star+\ell}$, where $\ell \in \{0, \ldots, k - k^\star\}$, encodes the following information: the existence of a monomial x^t in $T_{k^\star+\ell}$ implies that there is a subset $X \subseteq V(G)$ such that $t = \chi(X)$ and X is the union of $k^\star + \ell$ districts in which c_1 wins in $k^\star$ districts and every other candidate wins in at most $k^\star - 1$ districts. Therefore, it follows that if T_k contains the monomial $y^{\chi(V(G))}$ (the all 1-vector) then our algorithm should return "Yes", otherwise it should return "No". We define $T_{k^\star+\ell}$ recursively, with the base case given by $T_{k^\star} = \sum_{s=k^\star}^{n} Q_{1,(k^\star-1)}^s$. If $T_{k^\star} = 0$, then we return "No". We initialize $T_{k^\star+\ell} = 0$, for each $\ell \in \{1, \ldots, k - k^\star\}$. For each $i \in \{2, \ldots, m\}$, we proceed as follows in the increasing order of i.

– For each $j \in \{1, \ldots, \min\{k^\star - 1, k - k^\star\}\}$
  • For each $\ell \in \{j, \ldots, k - k^\star\}$ and $s \in \{k^\star + 1, \ldots, n\}$
    * Compute the polynomial $Q_\ell^s = \sum_{\substack{1 \leq s', s'' \leq s \\ s' + s'' = s}} P_i^{s'} \times \mathcal{H}_{s''}(T_{k^\star + \ell - 1})$
    * Compute the Hamming projection of Q_ℓ^s to s, that is, $Q_\ell^s = \mathcal{H}_s(Q_\ell^s)$
  • For each $\ell \in \{j, \ldots, k - k^\star\}$
    * Set $T_{k^\star + \ell} = \mathcal{R}(T_{k^\star + \ell} + \sum_{s=k^\star+1}^{n} Q_\ell^s)$

The range of j is dictated by the fact that since c_1 wins $k^\star$ districts, all other candidates combined can only win $k - k^\star$ districts and each individually may only win at most $k^\star - 1$ districts. Thus, overall candidate c_i, for any $i \geq 2$ can win at most $\min\{k^\star - 1, k - k^\star\}$ districts. The range of ℓ is dictated by the fact that (assuming that first $k^\star$ districts are won by c_1) j^{th} district won by c_i is either $(k^\star + j)^{\text{th}}$ district, or $(k^\star + j + 1)^{\text{th}}$ district, ..., or k^{th} district. The range of s is dictated by the fact that the number of vertices in the union of all the districts is at least $k^\star + 1$ as c_1 wins $k^\star$ districts.

Note that Q_ℓ^s is a non-zero polynomial if there exists a subset of vertices of size s that are formed by the union of $k^\star + \ell$ pairwise disjoint districts, $k^\star$ of which are won by c_1 and every other candidate wins at most $k^\star - 1$. Thus, the recursive definition of $T_{k^\star + \ell}$ is self explanatory. Next, we prove the correctness and running time of the algorithm which conclude the proof of Theorem 1.

Correctness. The following lemma proves the completeness of the algorithm.

Lemma 2. *If* $(G, \mathcal{C}, \{w_v \colon \mathcal{C} \to \mathbb{Z}^+\}_{v \in V(G)}, p, k, k^\star)$ *is a* Yes*-instance of* TW-GM *under a tie-breaking rule, then the above algorithm returns "*Yes*".*

Proof. Suppose that $V_1, \ldots, V_k$ is a solution to $(G, \mathcal{C}, \{w_v \colon \mathcal{C} \to \mathbb{Z}^+\}_{v \in V(G)}, p, k, k^\star)$. Recall that we assumed that $p = c_1$. Let $\mathcal{V}_i \subseteq \{V_1, \ldots, V_k\}$ be the set of districts won by the candidate c_i. Due to the application of a tie-breaking rule, $\mathcal{V}_i$s are pairwise disjoint. Without loss of generality, let $\mathcal{V}_1 = \{V_1, \ldots, V_{k^\star}\}$. We begin with the following claim that enables us to conclude that polynomial T_k has monomial $y^{\chi(V(G))}$.

Claim 1 (♣). *For each* $i \in \{1, \ldots, m\}$, *polynomial* $T_{\sum_{|\mathcal{V}_1| + \ldots + |\mathcal{V}_i|}}$ *contains the monomial* $y^{\chi(\cup_{Y \in \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_i} Y)}$.

Hence, we can conclude that the polynomial T_k contains the monomial $y^{\chi(V(G))}$. Hence, the algorithm returns Yes. □

In the next lemma, we prove the soundness of the algorithm.

Lemma 3. *If the above algorithm returns "*Yes*" for an instance* $I = (G, \mathcal{C}, \{w_v \colon \mathcal{C} \to \mathbb{Z}^+\}_{v \in V(G)}, c_1, k, k^\star)$ *for the tie-breaking rule* η, *then* I *is a* Yes*-instance of* TW-GM *under the tie-breaking rule* η.

Proof. We first prove the following claims.

Claim 2 (♣). *If $T_{k^\star}$ has a monomial y^S, then there are $k^\star$ pairwise disjoint districts $Y_1, \ldots, Y_{k^\star}$ such that $\chi(Y_1 \cup \ldots \cup Y_{k^\star}) = S$ and c_1 wins in every district.*

Claim 3 (♣). *For a pair of integer i, j, where $i \in \{2, \ldots, m\}$ and $j \in \{1, \ldots, \min\{k - k^\star, k^\star - 1\}\}$, let $T_{k^\star+1}, \ldots, T_k$ be the family of polynomials constructed in the algorithm at the end of for loops for i and j, in the above algorithm. Let y^S be a monomial in T_t, where $t \in \{k^\star+1, \ldots, k\}$. Then, the following hold:*

- *there are t pairwise disjoint districts $Y_1, \ldots, Y_t$ such that $\chi(Y_1 \cup \ldots \cup Y_t) = S$*
- *c_1 wins in $k^\star$ districts in $\{Y_1, \ldots, Y_t\}$*
- *c_i wins in j districts in $\{Y_1, \ldots, Y_t\}$*
- *for $2 \leq q < i$, c_q wins in at most $k^\star - 1$ districts in $\{Y_1, \ldots, Y_t\}$*
- *for $q > i$, c_q does not win in any district in $\{Y_1, \ldots, Y_t\}$*

The proof of this claim follows by using nested induction on i and j. If the algorithm returns Yes, then we know that there is a monomial $y^{\chi(V(G))}$ in T_k. Therefore, due to Claim 3, there are k districts such that c_1 wins in $k^\star$ districts and all the candidates win in at most $k^\star - 1$ districts. □

Lemma 4 (♣). *The above algorithm runs in $2^n(n+m)^{\mathcal{O}(1)}$ time.*

4 Deterministic Algorithm for Path

In this section, we discuss the proof of Theorem 2, the full details of each proof is in the Appendix. We note that due to Lemma 1, it is sufficient to present a deterministic FPT algorithm parameterized by k for TW-GM when the input is a path. Let $(G, \mathcal{C}, \{w_v : \mathcal{C} \to \mathbb{Z}^+\}_{v \in V(G)}, p, k, k^\star)$ be the input instance where G is the path $(u_1, \ldots, u_n)$. We begin with a simple observation.

Observation 2 *A path G on n vertices has $\mathcal{O}(n^2)$ distinct connected sets.*

Based on the above observation we create an auxiliary directed graph H with parallel arcs on $\binom{n}{2} + n + 2$ vertices, where we have a vertex for each connected set of G. For $\{i, j\} \subseteq [n]$, $i \leq j$, let $P_{i,j}$ denote the subpath of G starting at the i^{th} vertex and ending at the j^{th} vertex. That is $P_{i,j}$ is the subpath $(u_i, \ldots, u_j)$ of G. Formally, we define the auxiliary graph H as follows.

1. For each $\{i, j\} \subseteq \{1, \ldots, n\}$ such that $i \leq j$, create a vertex $v_{i,j}$ corresponding to the subpath $P_{i,j}$. 2. We do the following for each $\{i, j\} \subseteq \{1, \ldots, n\}$. Let c denote the candidate that wins the district $P_{i,j}$, where $i \leq j$. If $c \neq p$, then we do the following. For each $r \in \{j+1, \ldots, n\}$, we add $k^\star - 1$ arcs $\langle v_{i,j}, v_{j+1,r}, 1\rangle, \langle v_{i,j}, v_{j+1,r}, 2\rangle, \ldots, \langle v_{i,j}, v_{j+1,r}, k^\star - 1\rangle$ from vertex $v_{i,j}$ to $v_{j+1,r}$. We label the $k^\star - 1$ arcs from $v_{i,j}$ to $v_{j+1,r}$ with $\langle c, 1\rangle, \langle c, 2\rangle, \ldots, \langle c, k^\star - 1\rangle$. That is, for each $k' \in \{1, \ldots, k^\star - 1\}$, the arc $\langle v_{i,j}, v_{j+1,r}, k'\rangle$ is labeled with $\langle c, k'\rangle$. If $c = p$, then we do the following. For each $r \in \{j+1, \ldots, n\}$, we add an unlabeled

arc from $v_{i,j}$ to $v_{j+1,r}$. 3. Finally, we add two new vertices s and t. Now we add arcs incident to s. Let $i \in \{1, \ldots, n\}$. We add an unlabeled arc from the vertex s to $v_{1,i}$. Next we add arcs incident to t. Let c denote the candidate that wins in $P_{i,n}$. If $c \neq p$, then we add $k^\star - 1$ arcs $\langle v_{i,n}, t, 1\rangle, \langle v_{i,n}, t, 2\rangle, \ldots, \langle v_{i,n}, t, k^\star - 1\rangle$ from $v_{i,n}$ to t and label them with $\langle c, 1\rangle, \langle c, 2\rangle, \ldots, \langle c, k^\star - 1\rangle$, respectively. If $c = p$, then we add an unlabeled arc from $v_{i,n}$ to t.

Lemma 5 (♣). *There is a path on $k+2$ vertices from s to t in H such that the path has $k - k^\star$ labeled arcs with distinct labels and $k^\star + 1$ unlabeled arcs if and only if $V(G)$ can be partitioned into k districts such that p wins in $k^\star$ districts and any other candidate wins in at most $k^\star - 1$ districts.*

Thus, our problem reduces to finding a path on $k+2$ vertices from s to t in H such that there are $k^\star + 1$ unlabeled arcs, and $k - k^\star$ distinctly labeled arcs.

Theorem 5. *There is an algorithm that given an instance $\mathcal{I}$ of* TW-GM *and a tie-breaking rule, solves the instance $\mathcal{I}$ in time $2.619^{k-k^\star}|\mathcal{I}|^{\mathcal{O}(1)}$.*

Towards proving Theorem 5, we design a dynamic programming algorithm using the concept of *representative family*. We first define representative family.

Let $\mathcal{S}$ be a family of subsets of a universe U; and let $q \in \mathbb{N}$. A subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is said to *q-represent* $\mathcal{S}$ if the following holds. For every set B of size q, if there is a set $A \in \mathcal{S}$ such that $A \cap B = \emptyset$, then there is a set $A' \in \widehat{\mathcal{S}}$ such that $A' \cap B = \emptyset$. If $\widehat{\mathcal{S}}$ q-represents $\mathcal{S}$, then we call $\widehat{\mathcal{S}}$ a *q-representative of* $\mathcal{S}$.

Proposition 3. [21] *Let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be a family of sets of size p over a universe of size n and let $0 < x < 1$. For a given $q \in \mathbb{N}$, a q-representative family $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ for $\mathcal{S}$ with at most $x^{-p}(1-x)^{-q} \cdot 2^{o(p+q)}$ sets can be computed in time $\mathcal{O}((1-x)^{-q} \cdot 2^{o(p+q)} \cdot t \cdot \log n)$.*

We introduce the definition of subset convolution on set families which will be used to capture the idea of "extending" a partial solution, a central concept when using representative family. For two families of sets $\mathcal{A}$ and $\mathcal{B}$, we define $\mathcal{A} * \mathcal{B}$ as $\{A \cup B \colon A \in \mathcal{A}, B \in \mathcal{B}, A \cap B = \emptyset\}$.

Proof (Proof sketch of Theorem 5). An instance of TW-GM is given by $\mathcal{I} = (G, \mathcal{C}, \{w_v\}_{v \in V}, p, k, k^\star)$. Additionally, recall the construction of the labeled digraph H with parallel arcs from $\mathcal{I}$. In order to prove Theorem 5, due to Lemma 5, it is enough to decide whether there exists a path on $k+2$ vertices from s to t in H that satisfies the following properties: **(PI)** there are $k^\star + 1$ unlabeled arcs, and **(PII)** the remaining $k - k^\star$ arcs have distinct labels.

Before presenting our algorithm, we first define some notations. For $i \in \{1, \ldots, k+1\}$ and $r \in \{1, \ldots, k^\star + 1\}$, a path P starting from s on $i+1$ vertices is said to satisfy $\mathscr{P}(i, r)$ if there are r unlabeled arcs (including the arc from s in P), and the remaining $i - r$ arcs have distinct labels. For a subgraph H' of H, we denote the set of labels in the graph H' by $\mathcal{L}(H')$. Recall that each vertex $v \in V(H) \setminus \{s, t\}$ corresponds to a subpath (i.e., a district) of the path

G. Hence, for each $v \in V(H) \setminus \{s,t\}$, we use $\text{win}(v)$ to denote the (unique) candidate that wins[1] the district denoted by v. Equivalently, we say that the candidate $\text{win}(v)$ *wins* the district v in G. For each vertex $v \in V(H)$, and a pair of integers $i \in \{1,\ldots,k+1\}$, $r \in \{1,\ldots,\min\{i,k^\star+1\}\}$, we define a set family $\mathscr{F}[i,r,v] = \{P : P \textit{ is a } s \textit{ to } v \textit{ path in } H \textit{ on } i{+}1 \textit{ vertices satisfying } \mathscr{P}(i,r)\}$.

The following family contains the arc labels on the path in the family $\mathscr{F}[i,r,v]$. $\mathscr{Q}[i,r,v] = \big\{\mathcal{L}(P)\colon P \in \mathscr{F}[i,r,v]\big\}$. Note that for each value of $i \in \{1,\ldots,k+1\}$, r defined above and $v \in V(H)$, each set in $\mathscr{Q}[i,r,v]$ is actually a subset of $\mathcal{L}(H)$ of size $i-r$. If there is a path from s to t on $k+2$ vertices with $k-k^\star$ arcs with distinct labels, then $\mathscr{Q}[k+1,k^\star+1,t] \neq \emptyset$ and vice versa. That is, $\mathscr{Q}[k+1,k^\star+1,t] \neq \emptyset$ if and only if $\mathscr{F}[k+1,k^\star+1,t] \neq \emptyset$. Hence, to solve our problem, it is sufficient to check if $\mathscr{Q}[k+1,k^\star+1,t]$ is non-empty. To decide this, we design a dynamic programming algorithm using representative families over $\mathcal{L}(H)$. In this algorithm, for each value of $i \in \{1,\ldots,k+1\}$, $r \in \{1,\ldots,\min\{i,k^\star+1\}\}$, and $v \in V(H)$, we compute a $(k-k^\star-(i-r))$ representative family of $\mathscr{Q}[i,r,v]$, denoted by $\widehat{\mathscr{Q}}[i,r,v]$, using Proposition 3, where $x = \frac{i-r}{2(k-k^\star)-(i-r)}$. Here, the value of x is set with the goal to optimize the running time of our algorithm, as is the case for the algorithm for k-Path in [21]. Our algorithm outputs "Yes" if and only if $\widehat{\mathscr{Q}}[k+1,k^\star+1,t] \neq \emptyset$.

Algorithm. We now formally describe how we recursively compute the family $\widehat{\mathscr{Q}}[i,r,v]$, for each $i \in \{1,\ldots,k+1\}$, $r \in \{1,\ldots,\min\{i,k^\star+1\}\}$, and $v \in V(H)$.

Base Case: We set $\widehat{\mathscr{Q}}[1,r,v] = \mathscr{Q}[1,r,v]$

$$= \begin{cases} \{\emptyset\} & \text{if } \langle s,v\rangle \text{ is an arc in } H \text{ and } r=1 \\ \emptyset & \text{otherwise} \end{cases} \tag{1}$$

For each $i \in \{1,\ldots,k+1\}$, $r \in \{1,\ldots,k-k^\star\} \cup \{0\}$, and $v \in V(H)$, we set

$$\widehat{\mathscr{Q}}[i,r,v] = \mathscr{Q}[i,r,v] = \emptyset \text{ if } r = 0 \text{ or } r > i. \tag{2}$$

We define (2) so that the recursive definition (3) has a simple description.

Recursive Step: For each $i \in \{2,\ldots,k+1\}$, $r \in \{1,\ldots,\min\{i,k^\star+1\}\}$, and $v \in V(H)$, we compute $\widehat{\mathscr{Q}}[i,r,v]$ as follows. We first compute $\mathscr{Q}'[i,r,v]$ from the previously computed families and then we compute a $(k-k^\star-(i-r))$-representative family $\widehat{\mathscr{Q}}[i,r,v]$ of $\mathscr{Q}'[i,r,v]$. The family $\mathscr{Q}'[i,r,v]$ is computed using the representative family as follows: $\mathscr{Q}'[i,r,v] =$

$$\Big(\bigcup_{\substack{w\in N^-(v),\\ \text{win}(w)=p}} \widehat{\mathscr{Q}}[i-1,r-1,w]\Big) \bigcup \Big(\bigcup_{\substack{w\in N^-(v),\\ \text{win}(w)\neq p}} \widehat{\mathscr{Q}}[i-1,r,w] * \{\{\langle \text{win}(w), j\rangle\} : 1 \leq j < k^\star\}\Big) \tag{3}$$

Next, we compute a $(k-k^\star-(i-r))$-representative family $\widehat{\mathscr{Q}}[i,r,v]$ of $\mathscr{Q}'[i,r,v]$ using Proposition 3, where $x = \frac{i-r}{2(k-k^\star)-(i-r)}$. Our algorithm works as

[1] We may assume this by applying the tie-breaking rule.

follows: compute $\widehat{\mathscr{Q}}[i,r,v]$ using Eqs. (1)–(3), and Proposition 3. Output "Yes" if and only if $\widehat{\mathscr{Q}}[k+1,k^\star+1,t] \neq \emptyset$.

Correctness Proof. We prove that for every $i \in \{1,\ldots,k+1\}$, $r \in \{1,\ldots,\min\{i,k^\star+1\}\}$, and $v \in V(H)$, $\widehat{\mathscr{Q}}[i,r,v]$ is indeed a $(k-k^\star-(i-r))$ representative family of $\mathscr{Q}[i,r,v]$, and not just that of $\mathscr{Q}'[i,r,v]$. From the definition of 0-representative family of $\mathscr{Q}[k+1,k^\star+1,t]$, we have that $\mathscr{Q}[k+1,k^\star+1,t] \neq \emptyset$ if and only if $\widehat{\mathscr{Q}}[k+1,k^\star+1,t] \neq \emptyset$. Thus, for correctness we prove the following.

Lemma 6 (♣). *For each $i \in \{1,\ldots,k+1\}$, $r \in \{1,\ldots,\min\{i,k^\star+1\}\}$, and $v \in V(H)$, family $\widehat{\mathscr{Q}}[i,r,v]$ is a $(k-k^\star-(i-r))$-representative of $\mathscr{Q}[i,r,v]$.*

We first prove that the following recurrence for $\mathscr{Q}[i,r,v]$ is correct. $\mathscr{Q}[i,r,v] =$

$$\Big(\bigcup_{\substack{w\in N^-(v),\\ \mathrm{win}(w)=p}} \mathscr{Q}[i-1,r-1,w]\Big) \bigcup \Big(\bigcup_{\substack{w\in N^-(v),\\ \mathrm{win}(w)\neq p}} \mathscr{Q}[i-1,r,w] * \{\{\langle \mathrm{win}(w), j\rangle\} : 1 \leq j < k^\star\}\Big) \quad (4)$$

We claim that Eqs. (1), (2), and (4) correctly compute $\mathscr{Q}[i,r,v]$, for each $i \in \{1,\ldots,k+1\}$, $r \in \{1,\ldots,\min\{i,k^\star+1\}\}$, and $v \in V(H)$. This concludes the proof of Lemma 6 by showing subset containment on both sides. □

5 In Conclusion

We have shown that GM on paths is NP-complete, thereby resolving an open question in [24]. This gives parameterized intractability for parameters such as maximum degree of a vertex in the graph. Furthermore, we have presented FPT algorithms for paths when parameterized by the number of districts. We also give an FPT algorithm running in time $2^n(n+m)^{\mathcal{O}(1)}$ on general graphs.

We conclude with a few directions for further research: (i) Does there exist a $\mathcal{O}(c^n)$ algorithm for W-GM when there are possibly multiple winners in a district?; (ii) Is W-GM on paths FPT parameterized by the number of candidates?; (iii) Is W-GM on trees FPT parameterized by the number of districts?

References

1. Bentert, M., Koana, T., Niedermeier, R.: The complexity of gerrymandering over graphs: paths and trees. arXiv preprint arXiv:2102.08905 (2021)
2. Betzler, N., Guo, J., Niedermeier, R.: Parameterized computational complexity of Dodgson and Young elections. Inf. Comput. **208**(2), 165–177 (2010)
3. Betzler, N., Uhlmann, J.: Parameterized complexity of candidate control in elections and related digraph problems. Theor. Comput. Sci. **410**(52), 5425–5442 (2009)
4. Bevern, R.V., Bredereck, R., Chen, J., Froese, V., Niedermeier, R., Woeginger, G.J.: Network-based vertex dissolution. SIDMA **29**(2), 888–914 (2015)
5. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Narrow sieves for parameterized paths and packings. J. Comput. Syst. Sci. **87**, 119–139 (2017)

6. Brubach, B., Srinivasan, A., Zhao, S.: Meddling metrics: the effects of measuring and constraining partisan gerrymandering on voter incentives. In: Proceedings of EC 2020, pp. 815–833 (2020)
7. Clough, E.: Talking locally and voting globally: Duverger's law and homogeneous discussion networks. Political Res. Q. **3**(60), 531–540 (2007)
8. Cohen-Zemach, A., Lewenberg, Y., Rosenschein, J.S.: Gerrymandering over graphs. In: Proceedings of AAMAS 2018, pp. 274–282 (2018)
9. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
10. Cygan, M., Pilipczuk, M.: Exact and approximate bandwidth. Theor. Comput. Sci. **411**(40–42), 3701–3713 (2010)
11. Dey, P.: Gerrymandering: a briber's perspective. arXiv:1909.01583 (2019)
12. Dey, P., Misra, N., Narahari, Y.: Parameterized dichotomy of choosing committees based on approval votes in the presence of outliers. Theor. Comput. Sci. **783**, 53–70 (2019)
13. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer, Heidelberg (2012)
14. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer, London (2013). https://doi.org/10.1007/978-1-4471-5559-1
15. Eiben, E., Fomin, F.V., Panolan, F., Simonov, K.: Manipulating districts to win elections: fine-grained complexity. In: Proceedings of AAAI 2020 (2020)
16. Ellenberg, J.: How computers turned gerrymandering into a science. New York Times, October 2017
17. Erikson, R.S.: Malapportionment, gerrymandering, and party fortunes in congressional elections. Am. Political Sci. Rev. **4**(66), 1234–1245 (1972)
18. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Copeland voting fully resists constructive control. In: Fleischer, R., Xu, J. (eds.) AAIM 2008. LNCS, vol. 5034, pp. 165–176. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68880-8_17
19. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Llull and Copeland voting computationally resist bribery and constructive control. JAIR **35**, 275–341 (2009)
20. Fleiner, B., Nagy, B., Tasnádi, A.: Optimal partisan districting on planar geographies. Cent. Eur. J. Oper. Res. **25**(4), 879–888 (2017)
21. Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.: Efficient computation of representative families with applications in parameterized and exact algorithms. J. ACM **63**(4), 1–60 (2016)
22. Gupta, S., Jain, P., Panolan, F., Roy, S., Saurabh, S.: Gerrymandering on graphs: computational complexity and parameterized algorithms. arXiv preprint arXiv:2102.09889 (2021)
23. Issacharoff, S.: Gerrymandering and political cartels. Harvard Law Rev. **116**, 593–648 (2002)
24. Ito, T., Kamiyama, N., Kobayashi, Y., Okamoto, Y.: Algorithms for gerrymandering over graphs. In: Proceedings of AAMAS 2019 (2019)
25. Xia, L., Zuckerman, M., Procaccia, A.D., Conitzer, V., Rosenschein, J.S.: Complexity of unweighted coalitional manipulation under some common voting rules. In: Proceedings of IJCAI 2019 (2009)
26. Lewenberg, Y., Lev, O., Rosenschein, J.S.: Divide and conquer: using geographic manipulation to win district-based elections. In: Proceedings of AAMAS 2017 (2017)

27. Moenck, R.T.: Practical fast polynomial multiplication. In: Proceedings of SYMSAC 1976, pp. 136–148 (1976)
28. Monien, B.: How to find long paths efficiently. In: North-Holland Mathematics Studies, vol. 109, pp. 239–254. Elsevier (1985)
29. Neidermeier, R.: Invitation to Fixed-Parameter Algorithms. Springer (2006)
30. Puppe, C., Tasnádi, A.: Optimal redistricting under geographical constraints: why "pack and crack" does not work. Econ. Lett. **105**(1), 93–96 (2009)
31. Talmon, N.: Structured proportional representation. Theor. Comput. Sci. **708**, 58–74 (2018)
32. Williams, R.: Finding paths of length k in $O^\star(2^k)$ time. Inf. Process. Lett. **109**(6), 315–318 (2009)
33. Wines, M.: What is gerrymandering? And how does it work? New York Times, June 2019
34. Zuckerman, M., Procaccia, A.D., Rosenschein, J.S.: Algorithms for the coalitional manipulation problem. JAIR **173**(2), 392–412 (2009)